Необычные случаи оптимизации производительности на примере ClickHouse





Обо мне

Алексей, разработчик ClickHouse.



ClickHouse

- не тормозил;
- не тормозит;
- будет не тормозить ещё больше.



Оптимизация производительности

Профилирование на разных нагрузках.

Оптимизация всего, что вылезает.

Про тестирование производительности — смотрите доклад Александра Кузьменкова завтра в 10 утра.



https://www.techdesignforums.com/practice/technique/winning-at-whac-a-mole-redesigning-an-rf-transceiver/



Эпизод 1: MergeTree vs Memory

В ClickHouse есть разные «движки таблиц».

MergeTree таблицы хранят данные на диске.

Memory таблицы хранят данные в оперативке.

Память быстрее, чем диски*.

Значит Memory таблицы быстрее, чем MergeTree?

* Что значит «быстрее»?. Скорость последовательного чтения и записи. Задержки случайных чтений и записи. IOPS при заданном параллелизме и распределении нагрузки.

Конечно память может быть медленнее, чем дисковая подсистема, например одноканальная память vs. 10x PCIe 4.0 SSDs.





pictures : mc — Konsole

MergeTree vs Memory

Memory таблицы хранят данные в оперативке.

MergeTree таблицы хранят данные на диске,
точнее в файловой системе.

Но данные из файловой системы попадают в **page cache**.

И затем читаются уже из оперативки.

Значит нет разницы между **Memory** и **MergeTree** таблицами
в случае наличия данных в page cache?



MergeTree vs Memory

Очевидные случаи, когда **MergeTree** быстрее, чем **Memory**.

MergeTree таблицы имеют первичный ключ и вторичные индексы,
и позволяют читать только нужные диапазоны данных.

Memory таблицы позволяют только full scan.

Но этот случай не интересен.

А при full scan может MergeTree быть быстрее, чем Memory?



~ : mc — Konsole

File Edit View Bookmarks Settings Help

```
| ст 12.168.0.1              | http://sexstant;}p.captcha?p=1&text
           |     3567 |
| orton                      | http://tankionliner.by/compani-8-seriildi
           |     2402 |
| игры лица и гым чан дизайн | http://korer.ru/Session/team/sondakika/gruzki-svjazhene/seasonvali-goda.yandex.do?id=197&img
_url=htt |     2166 |
| imgsrc                     | http://haberi.hurriyet.com/bets_z_krymskih-porno/438828f3c987176434229589%2FarchPeriod=60&wp
=1&text= |     1848 |
| брызговик                  | http://auto.ru/chatay-barana.ru/traction.html#maybettaya
           |     1659 |
| индийский афтозный         | http://korer.ru/categories.ru/?vkb
           |     1549 |
| оооootводка и              | http://hurpass
           |     1480 |
| выступная мужчин           | http://korer.ru/categories.ru/?vkb
           |     1247 |
| юность                     | http://images/inbox/?_1ld
           |     1112 |

10 rows in set. Elapsed: 0.221 sec. Processed 8.87 million rows, 880.65 MB (40.15 million rows/s., 3.98 GB/s.)

milovidov-desktop :) Bye.
milovidov@milovidov-desktop:~/work/clickhouse-presentations/highload2021/pictures$
milovidov@milovidov-desktop:~$ clickhouse-client
ClickHouse client version 21.6.1.1.
Connecting to localhost:9000 as user default.
Connected to ClickHouse server version 21.6.1 revision 54448.

milovidov-desktop :) 
```

~ : mc | highload2021 : mc | highload2018 : mc | ~ : bash | ~ : bash | ~ : bash | ~ : bash | ~ : sudo openvpn

index.html * — Kate

File Edit View Projects Bookmarks Sessions Tools Settings Help

highload2021 / index.html

```
        <p><b>MergeTree</b> таблицы имеют первичный ключ и вторичные индексы,<br/>и позволяют читать только нужные диапазоны данных.</p>
        <p><b>Memory</b> таблицы позволяют только full scan.</p>

        <p style="color: gray;">Но этот случай не интересен.</p>

        <p style="color: green;">А при full scan может MergeTree быть быстрее, чем Memory?</p>
    </section>

    <!-- clickhouse-benchmark <<< "SELECT SearchPhrase, any(replaceAll(substring(URL, 1, 100), 'google', 'yandex')) AS s, count() AS c FROM test.hits GROUP BY SearchPhrase ORDER BY c DESC LIMIT 10" -->

    <section class="slide">
        <video style="width: 100%;"><source src="video/memory_is_slower.ogv" type="video/ogg"></video>
    </section>
```

Find: #(\d+)

Replace: #\1

Mode: Regular expression | Replace | Replace All | Find All

Terminal | Search and Replace | Current Project

Konsole | Okular | index.html * — Kate | Telegram (9) | default: ClickHouse - [ClickHouse... | apt - 20.04: Can't install gtk-r... | Необычные случаи оптимизаци... | us 05:45



MergeTree vs Memory

Неочевидные случаи, когда **MergeTree** быстрее, чем **Memory**.

MergeTree таблицы хранят данные в сортированном порядке по первичному ключу.

Некоторые алгоритмы в ClickHouse эксплуатируют преимущества локальности данных, если она есть (fast path).

Например, если при GROUP BY подряд дважды встретилось одно и то же значение, то мы не делаем повторный поиск в хэш-таблице.

Про хэш-таблицы в ClickHouse смотрите доклад Максима Киты завтра в 12:50.

А если данные в таблицах находятся в одинаковом порядке, может ли MergeTree быть быстрее, чем Memory?



Как обрабатываются данные в ClickHouse

Данные в ClickHouse хранятся по столбцам
и обрабатываются тоже по столбцам.

Array of Structures

```
struct Point3d
{
    float x;
    float y;
    float z;
};
std::vector<Point3d> points;
```

Structure of Arrays

```
struct Points
{
    std::vector<float> x;
    std::vector<float> y;
    std::vector<float> z;
};
```



Как обрабатываются данные в ClickHouse

Данные в ClickHouse хранятся по столбцам
и обрабатываются тоже по столбцам. **По кусочкам столбцов**.

```
struct Chunk
{
    std::vector<float> x;
    std::vector<float> y;
    std::vector<float> z;
};

std::vector<Chunk> chunks;
```

— **Morsel-based processing**.



Как именно читаются данные из таблицы?

В случае **MergeTree**:

— читаем сжатые файлы из файловой системы;

— вычисляем и сверяем чексуммы;

— разжимаем сжатые блоки;

— десериализуем кусочки столбцов;

— обрабатываем их;



Это не оптимально



Как именно читаются данные из таблицы?



Это более оптимально

В случае **Memory**:

— в оперативке уже находятся готовые кусочки столбцов,

обрабатываем их;



Что именно происходит при чтении?

В случае **MergeTree**:

1. Читаем сжатые файлы из файловой системы:

— читать можно с помощью синхронного (**read**/**pread**, **mmap**) или асинхронного (**AIO**, **uring**) ввода-вывода;

— в случае синхронного ввода-вывода, можно использовать (обычный read или mmap) или не использовать page cache (**O_DIRECT**);

— если читать из page cache без mmap, то будет копирование из page cache в userspace;

— читаем сжатые данные — если коэффициент сжатия большой, то доля времени в обработке запроса маленькая;



Что именно происходит при чтении?

В случае **MergeTree**:

2. Разжимаем сжатые блоки:

— по-умолчанию используется LZ4*;

— можно выбрать как более сильный метод сжатия (ZSTD),
так и более слабый, например вообще без сжатия (NONE);

— иногда NONE внезапно работает медленнее, с чего бы это?

— а блоками какого размера были сжаты данные?
и как это влияет на скорость?

* Смотрите доклад «Как ускорить разжатие LZ4» с HighLoad++ Siberia 2018.



Что именно происходит при чтении?

В случае **MergeTree**:

3. Десериализуем кусочки столбцов:

— десериализации как таковой нет;

— это просто перекладывание данных (memcpy);

— а зачем вообще нужно перекладывать данные?



Отличие MergeTree и Memory

В случае **Memory**:
— готовые кусочки столбцов в оперативке.

В случае **MergeTree**:
— кусочки столбцов формируются динамически при чтении.

MergeTree делает больше работы,
но может ли это иногда быть оптимальнее?



MergeTree vs Memory

В случае **MergeTree**:

— кусочки столбцов формируются динамически при чтении, и их размер в числе строк может выбираться адаптивно для кэш-локальности!



Кэш-локальность

С какой скоростью работает оперативка?
— какая оперативка, на какой машине?

С какой скоростью работает кэш?
— кэш какого уровня, на каком CPU?
— один или все вместе?

С какой скоростью чего?
— throughput, latency?..



Эпизод 2: сжатие данных тормозит?

В ClickHouse данные по-умолчанию хранятся сжатыми.

При записи сжимаются, при чтении — разжимаются.

Профилируем запросы...

В топе по CPU — функция LZ4_decomress_safe.





HAHAHA, CLASSIC...

Megg, Mogg & Owl Series by Simon Hanselmann



Пробуем убрать сжатие данных

Но ничего хорошего из этого не выходит:



1. Убрали сжатие данных и теперь они не помещаются на диск.

2. Убрали сжатие данных и теперь чтение с диска тормозит.

3. Убрали сжатие данных и теперь меньше данных помещается в page cache.

...

Но даже если несжатые данные помещаются целиком в оперативку — **имеет ли смысл не сжимать** их?

Что быстрее: разжатие или memcpy?

Функцию **memcpy** используют как baseline
самого слабого сжатия или разжатия в бенчмарках.

Конечно, это самый быстрый эталон для сравнения.

Пример:
— memcpy: **12 ГБ** в секунду.
— LZ4 decompression: **2..4 ГБ** разжатых данных в секунду.

Вывод: memcpy быстрее, чем разжатие LZ4?



Что быстрее: разжатие или memcpy?

Рассмотрим сценарий:

— данные хранятся в оперативке;
— данные обрабатываются по блокам;
— каждый блок достаточно небольшой и помещается в кэш CPU;
— обработка каждого блока помещается в кэш CPU;
— **данные обрабатываются в несколько потоков**;

Данные читаются из оперативки, дальше используется только кэш CPU.



Что быстрее: разжатие или memcpy?

Пример: Ryzen 3950 (16 ядер)

— memcpy: 16×12 ГБ = **192 ГБ** в секунду.
— LZ4 decompression: 16×2..4 ГБ = **32..48 ГБ** разжатых данных в секунду.
— скорость чтения из памяти: **30 ГБ*** в секунду.

В случае memcpy чтение упирается в скорость памяти.

Но если используется сжатие, то из памяти читается меньше данных.
Память работает как диск. Разжатие LZ4 быстрее, чем memcpy?

* память двухканальная, но работает не на максимальной частоте.
По спецификации для этого CPU до 48 ГБ в секунду.



Что быстрее: разжатие или memcpy?

Пример: 2 × AMD EPYC 7742 (**128 ядер**)

8 channel memory, max throughput **190 GiB/s**

Для этого сервера работа с данными,
сжатыми LZ4, также будет быстрее.

Но если ядер меньше — уже не всё однозначно.

Если данные хорошо сжаты, то разжатие всё-таки упирается в CPU,
а значит, его можно ускорить!



Оптимизации в ClickHouse

Для **Memory** таблиц:

— Уменьшили размер блока при записи
для лучшей кэш-локальности обработки данных #20169.

— Возможность сжатия Memory таблиц #20168.

Для **MergeTree** таблиц:

— Убрали лишнее копирование для режима сжатия NONE #22145.

— Возможность отключить чексуммы при чтении #19588,
но использовать эту возможность не надо.

— Возможность чтения с помощью mmap #8520, чтобы убрать
лишнее копирование из page cache а также кэш memory mappings
#22206.



~ : mc — Konsole

File Edit View Bookmarks Settings Help

```
80.000%         0.218 sec.
90.000%         0.219 sec.
95.000%         0.219 sec.
99.000%         0.219 sec.
99.900%         0.219 sec.
99.990%         0.219 sec.


^CStopping launch of queries. SIGINT received.

Queries executed: 37.

localhost:9000, queries 37, QPS: 4.631, RPS: 41091849.686, MiB/s: 3889.060, result RPS: 46.306, result MiB/s: 0.004.

0.000%          0.208 sec.
10.000%         0.211 sec.
20.000%         0.211 sec.
30.000%         0.212 sec.
40.000%         0.214 sec.
50.000%         0.216 sec.
60.000%         0.216 sec.
70.000%         0.217 sec.
80.000%         0.219 sec.
90.000%         0.223 sec.
95.000%         0.225 sec.
99.000%         0.234 sec.
99.900%         0.234 sec.
99.990%         0.234 sec.


milovidov@milovidov-desktop:~$
```

~ : mc | video : mc | pictures : mc | ~ : bash | ~ : bash | ~ : bash | ~ : bash | ~ : sudo openvpn



~ : mc — Konsole

File Edit View Bookmarks Settings Help

```
┌─────────┬─────────c─┐
│                              │ http://image&uinfo/by_viddings
         │ 8267016 │
│ ст 12.168.0.1                │ http://korer.ru/moskva/regnum.ru/cars.auto.ru/video_haberyorum/shop.pl?cmd=show/1007/300x225
/8f905e6 │    3567 │
│ orton                        │ http://image&uinfo=ElsdCQBYRlRgDQZaHEVUXiN2A1FzBx5pU1BXIgZUJlIEA
         │    2402 │
│ игры лица и гым чан дизайн   │ http://korer.ru/Session/team/sondakika/gruzki-svjazhene/seasonvali-goda.yandex.ru/test.tanks
.ru/anke │    2166 │
│ imgsrc                       │ http://opel/other/frame/frm_index.com/iframeid=20500023249%26bt%3D62288586&CMD=-RR=0,0,0,0,s
ource=wi │    1848 │
│ брызговик                    │ http://image&uinformTypeBuyClick=http://plity=1&api_id=0&session_time=0&session_timeout=1440
&is_mobi │    1659 │
│ индийский афтозный           │ http://korer.ru/categories.ru/?vkb
         │    1549 │
│ ооооотводка и                │ http://avito.st%2Fattat/psycholog.shorttrackin_image
         │    1480 │
│ выступная мужчин             │ http://korer.ru/categories.ru/?vkb
         │    1247 │
│ юность                       │ http://images/inbox/?_1ld
         │    1112 │
└──────────────────────────────┴──────────────────────────────────────────────────────────────
─────────┴─────────┘

10 rows in set. Elapsed: 0.162 sec. Processed 8.87 million rows, 880.65 MB (54.89 million rows/s., 5.45 GB/s.)

milovidov-desktop :) Bye.
milovidov@milovidov-desktop:~$
milovidov@milovidov-desktop:~$
milovidov@milovidov-desktop:~$
milovidov@milovidov-desktop:~$
```

~ : mc | video : mc | pictures : mc | ~ : bash | ~ : bash | ~ : bash | ~ : bash | - : sudo openvpn



Выводы

Чтобы оптимизировать производительность, нужно всего лишь:

— точно знать, что делает ваш код;

— профилировать систему на реалистичных сценариях нагрузки;

— представлять возможности железа;

...

— не забывать что в системе много ядер, а у процессора есть кэш;
не путать latency и throughput :)



Спасибо!

